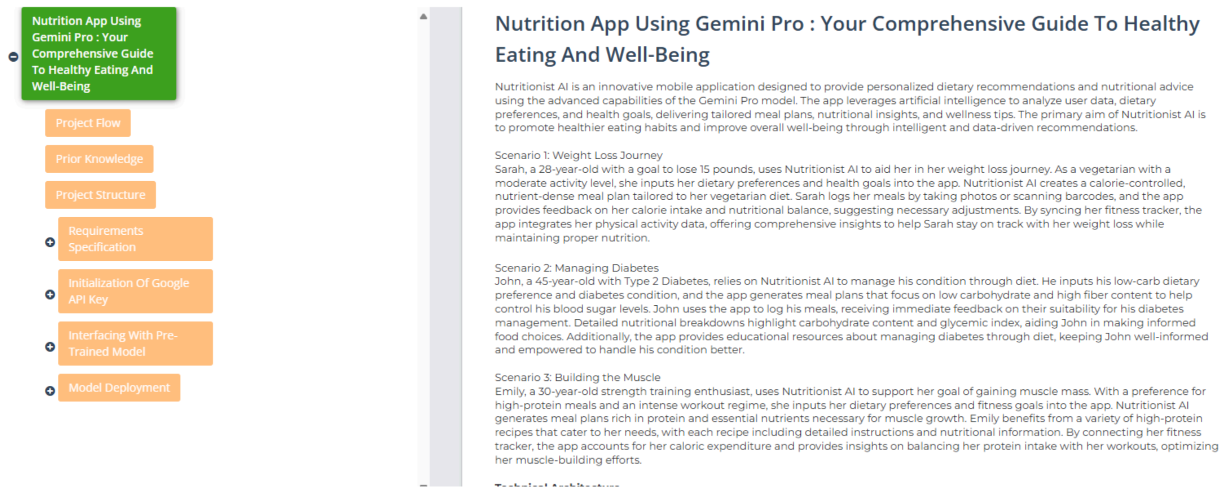
Gen AI App Health Management System

Created	@July 27, 2025 7:43 PM


Nutrition App Using Gemini Pro : Your Comprehensive Guide To Healthy Eating And Well-Being
Project Flow
Prior Knowledge
Project Structure
Requirements Specification
Initialization Of Google API Key
Interfacing With Pre-Trained Model
Model Deployment

Nutrition App Using Gemini Pro : Your Comprehensive Guide To Healthy Eating And Well-Being

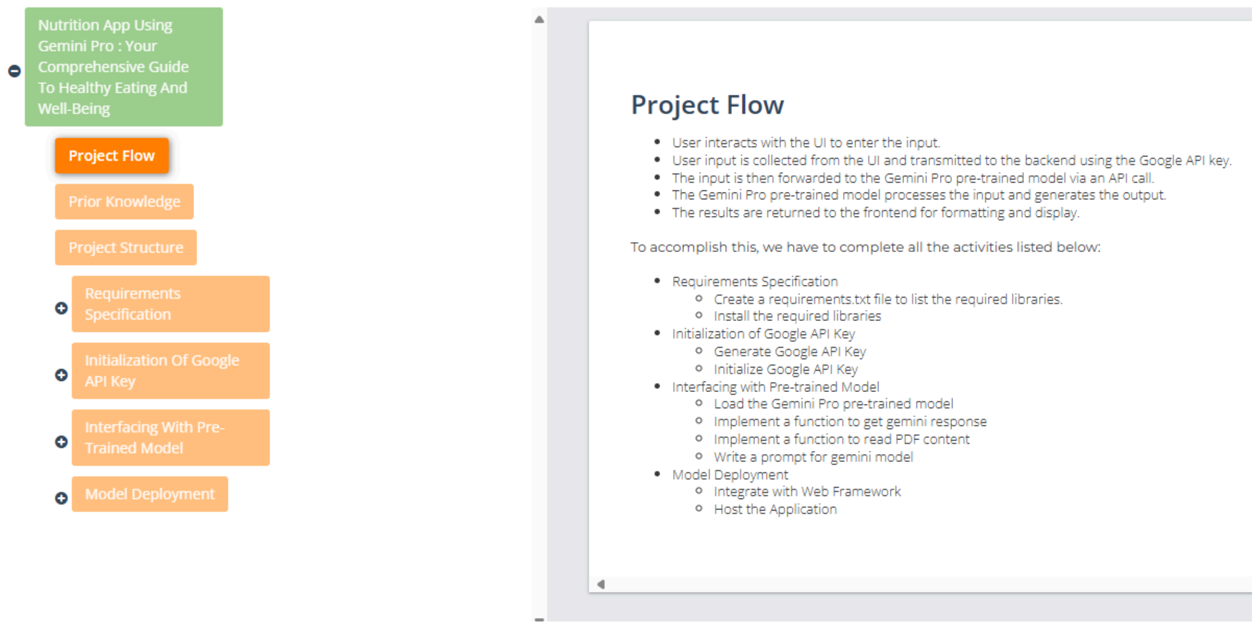
Nutritionist AI is an innovative mobile application designed to provide personalized dietary recommendations and nutritional advice using the advanced capabilities of the Gemini Pro model. The app leverages artificial intelligence to analyze user data, dietary preferences, and health goals, delivering tailored meal plans, nutritional insights, and wellness tips. The primary aim of Nutritionist AI is to promote healthier eating habits and improve overall well-being through intelligent and data-driven recommendations.

Scenario 1: Weight Loss Journey
Sarah, a 28-year-old with a goal to lose 15 pounds, uses Nutritionist AI to aid her in her weight loss journey. As a vegetarian with a moderate activity level, she inputs her dietary preferences and health goals into the app. Nutritionist AI creates a calorie-controlled, nutrient-dense meal plan tailored to her vegetarian diet. Sarah logs her meals by taking photos or scanning barcodes, and the app provides feedback on her calorie intake and nutritional balance, suggesting necessary adjustments. By syncing her fitness tracker, the app integrates her physical activity data, offering comprehensive insights to help Sarah stay on track with her weight loss while maintaining proper nutrition.

Scenario 2: Managing Diabetes
John, a 45-year-old with Type 2 Diabetes, relies on Nutritionist AI to manage his condition through diet. He inputs his low-carb dietary preference and diabetes condition, and the app generates meal plans that focus on low carbohydrate and high fiber content to help control his blood sugar levels. John uses the app to log his meals, receiving immediate feedback on their suitability for his diabetes management. Detailed nutritional breakdowns highlight carbohydrate content and glycemic index, aiding John in making informed food choices. Additionally, the app provides educational resources about managing diabetes through diet, keeping John well-informed and empowered to handle his condition better.

Scenario 3: Building the Muscle
Emily, a 30-year-old strength training enthusiast, uses Nutritionist AI to support her goal of gaining muscle mass. With a preference for high-protein meals and an intense workout regime, she inputs her dietary preferences and fitness goals into the app. Nutritionist AI generates meal plans rich in protein and essential nutrients necessary for muscle growth. Emily benefits from a variety of high-protein recipes that cater to her needs, with each recipe including detailed instructions and nutritional information. By connecting her fitness tracker, the app accounts for her caloric expenditure and provides insights on balancing her protein intake with her workouts, optimizing her muscle-building efforts.

This is the overview of the app.

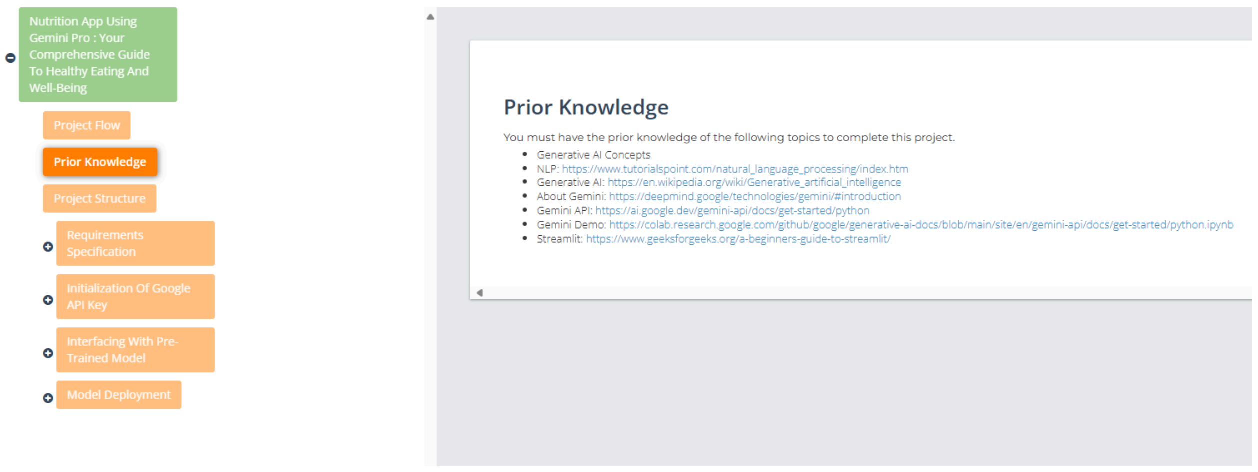

Nutrition App Using Gemini Pro : Your Comprehensive Guide To Healthy Eating And Well-Being
Project Flow
Prior Knowledge
Project Structure
Requirements Specification
Initialization Of Google API Key
Interfacing With Pre-Trained Model
Model Deployment

Project Flow

- User interacts with the UI to enter the input.
- User input is collected from the UI and transmitted to the backend using the Google API key.
- The input is then forwarded to the Gemini Pro pre-trained model via an API call.
- The Gemini Pro pre-trained model processes the input and generates the output.
- The results are returned to the frontend for formatting and display.

To accomplish this, we have to complete all the activities listed below:

- Requirements Specification
  - Create a requirements.txt file to list the required libraries.
  - Install the required libraries
- Initialization of Google API Key
  - Generate Google API Key
  - Initialize Google API Key
- Interfacing with Pre-trained Model
  - Load the Gemini Pro pre-trained model
  - Implement a function to get gemini response
  - Implement a function to read PDF content
  - Write a prompt for gemini model
- Model Deployment
  - Integrate with Web Framework
  - Host the Application


Nutrition App Using Gemini Pro : Your Comprehensive Guide To Healthy Eating And Well-Being
Project Flow
Prior Knowledge
Project Structure
Requirements Specification
Initialization Of Google API Key
Interfacing With Pre-Trained Model
Model Deployment

Prior Knowledge

You must have the prior knowledge of the following topics to complete this project.

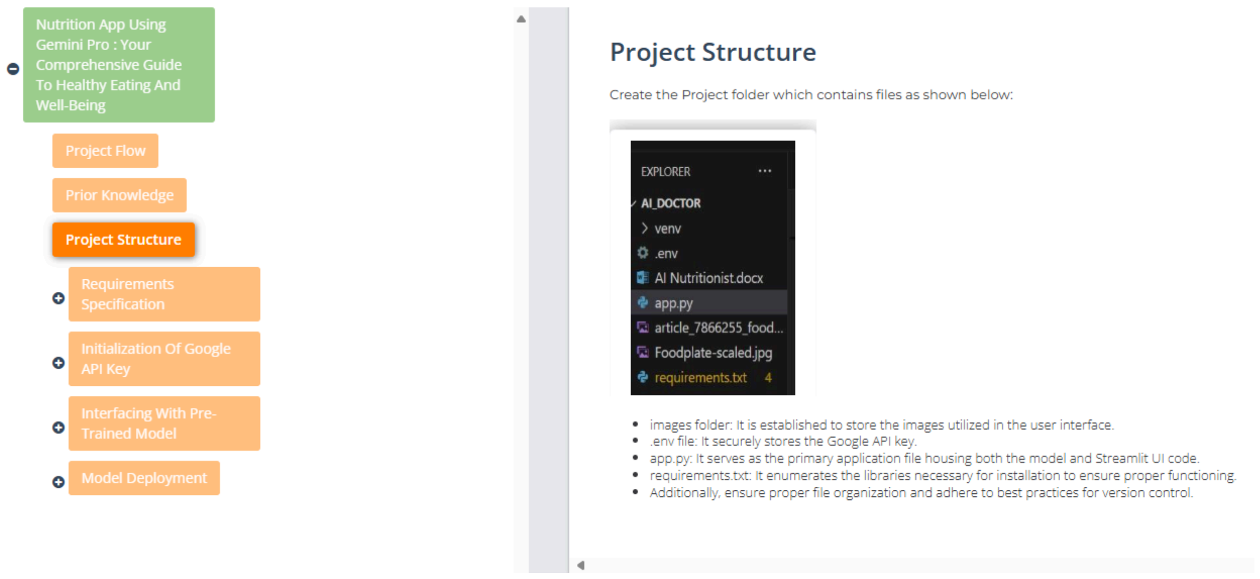
- Generative AI Concepts
- NLP: https://www.tutorialspoint.com/natural_language_processing/index.htm
- Generative AI: https://en.wikipedia.org/wiki/Generative_artificial_intelligence
- About Gemini: https://deepmind.google/technologies/gemini/#introduction
- Gemini API: https://ai.google.dev/gemini-api/docs/get-started/python
- Gemini Demo: https://colab.research.google.com/github/google/generative-ai-docs/blob/main/site/en/gemini-api/docs/get-started/python.ipynb
- Streamlit: https://www.geeksforgeeks.org/a-beginners-guide-to-streamlit/

The links given in this are-

- NLP: https://www.tutorialspoint.com/natural_language_processing/index.htm
- Generative AI: https://en.wikipedia.org/wiki/Generative_artificial_intelligence
- About Gemini: https://deepmind.google/technologies/gemini/#introduction
- Gemini API: https://ai.google.dev/gemini-api/docs/get-started/python

- Gemini Demo: https://colab.research.google.com/github/google/generative-ai-docs/blob/main/site/en/gemini-api/docs/get-started/python.ipynb
- Streamlit: https://www.geeksforgeeks.org/a-beginners-guide-to-streamlit/

Project Structure


Nutrition App Using Gemini Pro : Your Comprehensive Guide To Healthy Eating And Well-Being
Project Flow
Prior Knowledge
Project Structure
Requirements Specification
Initialization Of Google API Key
Interfacing With Pre-Trained Model
Model Deployment

Project Structure

Create the Project folder which contains files as shown below:

EXPLORER ...
AI_DOCTOR
venv
.env
AI Nutritionist.docx
app.py
article_7866255_food...
Foodplate-scaled.jpg
requirements.txt 4

- images folder: It is established to store the images utilized in the user interface.
- .env file: It securely stores the Google API key.
- app.py: It serves as the primary application file housing both the model and Streamlit UI code.
- requirements.txt: It enumerates the libraries necessary for installation to ensure proper functioning.
- Additionally, ensure proper file organization and adhere to best practices for version control.

▼ How To create an environment in python?

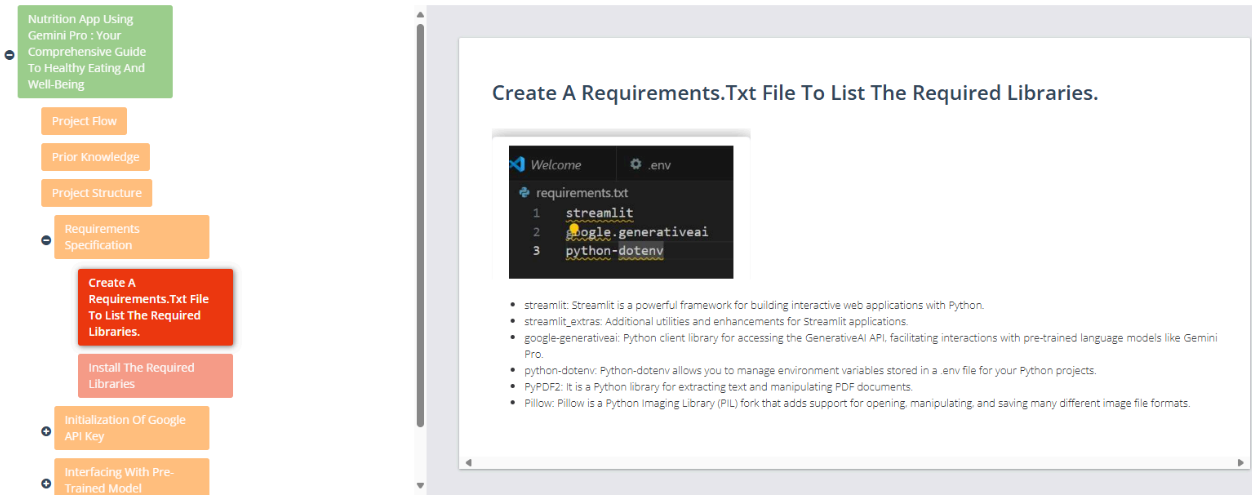
In explorer

- click on "..."
- Go to terminal
- click on "New Terminal" (it will open as power shell , we cannot create an env in powershell , only be created in command prompt)
- click on plus
- open cmd
- python -m venv <nameOfEnvironment>
- Give

▼ How to activate an environment? (venv)

nameOfEnv > Scripts > Activate

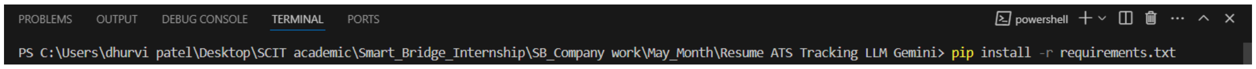
SYTNAX: nameOfEnv\Scripts\activate


Nutrition App Using Gemini Pro : Your Comprehensive Guide To Healthy Eating And Well-Being
Project Flow
Prior Knowledge
Project Structure
Requirements Specification
Create A Requirements.Txt File To List The Required Libraries.
Install The Required Libraries
Initialization Of Google API Key
Interfacing With Pre-Trained Model
Welcome
.env
requirements.txt

Create A Requirements.Txt File To List The Required Libraries.

```
streamlit
google.generativeai
python-dotenv
```

- streamlit: Streamlit is a powerful framework for building interactive web applications with Python.
- streamlit_extras: Additional utilities and enhancements for Streamlit applications.
- google-generativeai: Python client library for accessing the GenerativeAI API, facilitating interactions with pre-trained language models like Gemini Pro.
- python-dotenv: Python-dotenv allows you to manage environment variables stored in a .env file for your Python projects.
- PyPDF2: It is a Python library for extracting text and manipulating PDF documents.
- Pillow: Pillow is a Python Imaging Library (PIL) fork that adds support for opening, manipulating, and saving many different image file formats.



```
PS C:\Users\dhurvi patel\Desktop\SCIT academic\Smart_Bridge_Internship\SB_Company work\May_Month\Resume ATS Tracking LLM Gemini> pip install -r requirements.txt
```

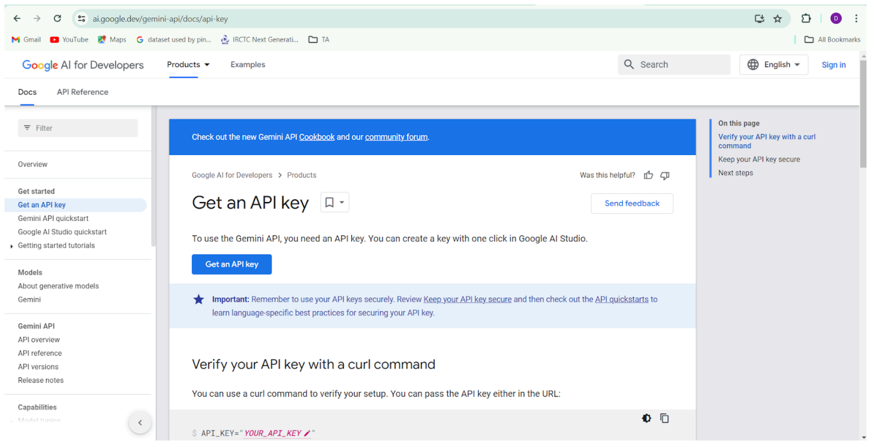
- Open the terminal.
- Run the command: pip install -r requirements.txt
- This command installs all the libraries listed in the requirements.txt file

Initialization of Google API Key

- [ ] **Generate Google API Key**

- Click the provided link to access the following webpage.

```
https://ai.google.dev/gemini-api/docs/api-key
```



Check out the new Gemini API Cookbook and our community forum.

Google AI for Developers > Products

Get an API key

To use the Gemini API, you need an API key. You can create a key with one click in Google AI Studio.

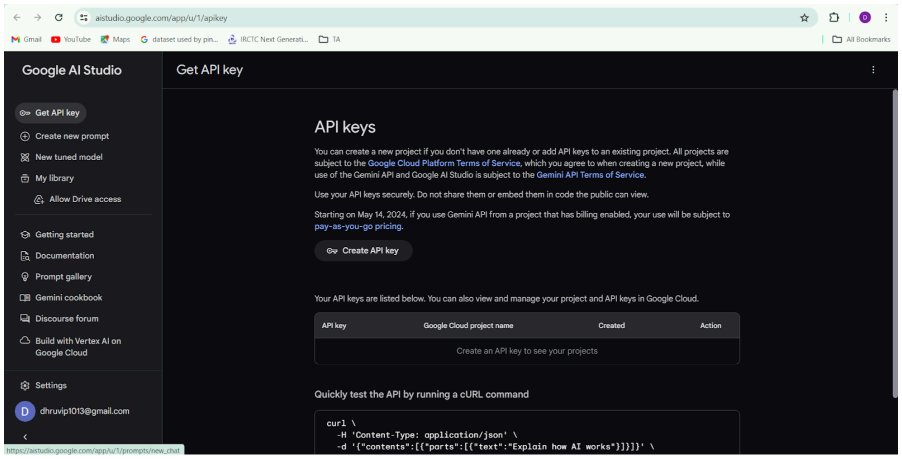
Important: Remember to use your API keys securely. Review Keep your API key secure and then check out the API quickstarts to learn language-specific best practices for securing your API key.

Verify your API key with a curl command

You can use a curl command to verify your setup. You can pass the API key either in the URL:

```
$ API_KEY="YOUR_API_KEY"
```

- After signing in to your account, navigate to the 'Get an API Key' option. Clicking on this option will redirect you to another webpage as shown below.



API keys

You can create a new project if you don't have one already or add API keys to an existing project. All projects are subject to the Google Cloud Platform Terms of Service, which you agree to when creating a new project, while use of the Gemini API and Google AI Studio is subject to the Gemini API Terms of Service.

Use your API keys securely. Do not share them or embed them in code the public can view.

Starting on May 14, 2024, if you use Gemini API from a project that has billing enabled, your use will be subject to pay-as-you-go pricing.

Your API keys are listed below. You can also view and manage your project and API keys in Google Cloud.

API key	Google Cloud project name	Created	Action
Create an API key to see your projects			

Quickly test the API by running a cURL command

```
curl \
  -H 'Content-Type: application/json' \
  -d '{"contents":[{"parts":[{"text":"Explain how AI works"}]}]}' \
```

- Next, click on 'Create API Key' and choose the generative language client as the project. Then, select 'Create API key in existing project'.
- Copy the newly generated API key as it is required for loading the Gemini Pro pre-trained model.

☐ Initialize Google API Key

```
GOOGLE_API_KEY="<Enter the copied Google API Key>"
```

- Create a .env file and define a variable named GOOGLE_API_KEY.

- Assign the copied Google API key to this variable.
- Paste the API key obtained from the previous steps here.

- [ ] **Load the Gemini Pro API**

```
###Health Management System
from dotenv import load_dotenv

load_dotenv()
import streamlit as st
import os
import google.generativeai as genai
from PIL import Image

genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))
```

This code snippet is for initializing a health management application using Streamlit, an open-source app framework, and Google Generative AI services. The script starts by loading environment variables from a .env file using the load_dotenv() function from the dotenv package. It then imports necessary libraries: streamlit for creating the web app interface, os for accessing environment variables, google.generativeai for utilizing Google's Generative AI capabilities, and PIL.Image for image processing. The genai.configure() function is called to set up the Google Generative AI API with the API key retrieved from the environment variables, ensuring secure and authorized access to the AI services.

STEP : Interfacing with Pre-trained Model

To interface with the pre-trained model, we'll start by creating an app.py file, which will contain both the model and Streamlit UI code.

☐ **Implement a function to get gemini response**

```
# Function to load Google Gemini Pro Vision API and get a response

def get_gemini_response(input, image, prompt):
    model = genai.GenerativeModel('gemini-1.5-flash')
    response = model.generate_content([input, image[0], prompt])
    return response.text
```

- The function get_gemini_response takes an input text as a parameter.
- It calls the generate_content method of the model object to generate a response.
- The generated response is returned as text.

☐ **Implement a function to read the Image and set the image format for Gemini Pro model Input**

```
# Function to handle the image input and prepare it for the API call

def input_image_setup(uploaded_file):
    # Check if a file has been uploaded
    if uploaded_file is not None:
        # Read the file into bytes
        bytes_data = uploaded_file.getvalue()

        # Prepare image data with mime type
        image_parts = [
            {
                "mime_type": uploaded_file.type,  # Get the mime type of the upload
ed file
                "data": bytes_data
            }
        ]
        return image_parts
    else:
        raise FileNotFoundError("No file uploaded")
```

The function input_image_setup processes an uploaded image file for a health management application. It first checks if a file has been uploaded. If a file is present, it reads the file's content into bytes and creates a dictionary containing the file's MIME type and its byte data. This dictionary is then stored in a list named image_parts, which is returned by the function. If no file is uploaded, the function raises a FileNotFoundError, indicating that an image file is required but not provided. This setup ensures that the uploaded image is correctly formatted and ready for further processing or analysis in the application.

STEP :Write a prompt for gemini model

```
# Input prompt for the AI model

input_prompt = """
You are an expert nutritionist. Your task is to analyze the food items visible in t
he uploaded image
and calculate the total calories. Also, provide the calorie details of each indivi
dual food item.

Use the following format:

1. Item 1 - number of calories
2. Item 2 - number of calories
...
Total Calories: <total>
"""
```

The variable input_prompt is a multi-line string designed as a prompt for a nutritionist AI model. It instructs the model to analyze an image of food items, identify each food item, and calculate the total calories. Additionally, the model is to provide a detailed breakdown of each food item with its respective calorie count. The expected output format is a numbered list where each item is listed alongside its calorie content, ensuring clarity and structured information for the user. This prompt is likely used in conjunction with an AI service that can process images and generate nutritional information based on the visual data provided.

STEP :Model Deployment

☐ **Integrate with Web Framework**

```
st.set_page_config(page_title="AI Nutritionist App")

# App title
st.header("AI Nutritionist App")

# Input prompt from the user
input = st.text_input("Input Prompt:", key="input")

# File uploader for images
uploaded_file = st.file_uploader("Choose an image...", type=["jpg", "jpeg", "pn
g"])

# Display the uploaded image
image = ""
if uploaded_file is not None:
    image = Image.open(uploaded_file)
    st.image(image, caption="Uploaded Image.", use_container_width=True)

# Submit button
submit = st.button("Tell me the total calories")
```

If “Tell me the total calories button click”:

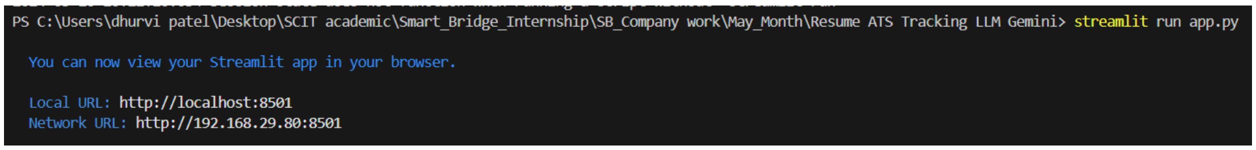
```
# If submit button is clicked
if submit:
    # Prepare the image data in required format
    image_data = input_image_setup(uploaded_file)

    # Get response from Gemini Vision model
    response = get_gemini_response(input_prompt, image_data, input)

    # Display the response
    st.subheader("The Response is:")
    st.write(response)
```

This code initializes a Streamlit application titled "AI Nutritionist App" by setting the page title and creating the app's header. It includes a text input field for users to enter a custom prompt and a file uploader for users to upload an image in JPG, JPEG, or PNG format. If an image is uploaded, it is opened using the PIL library and displayed within the app with a caption. A button labeled "Tell me the total calories" is also provided, which users can click to trigger the application's functionality for analyzing the uploaded image to calculate and display the total calorie content of the food items depicted.

STEP :Host the Application

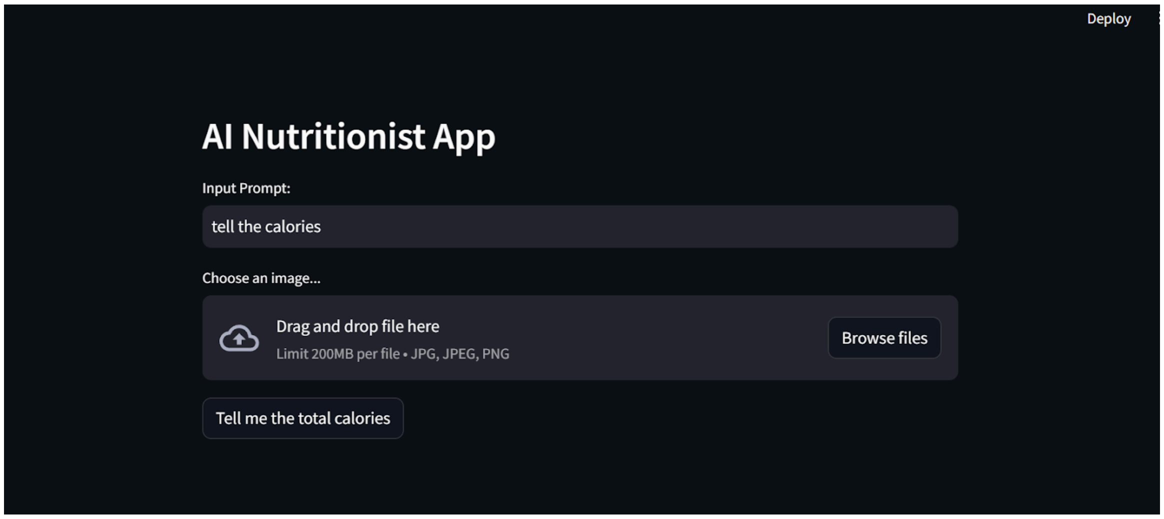
Launching the Application:

- To host the application, go to the terminal, type - streamlit run app.py
- Here app.py refers to a python script.

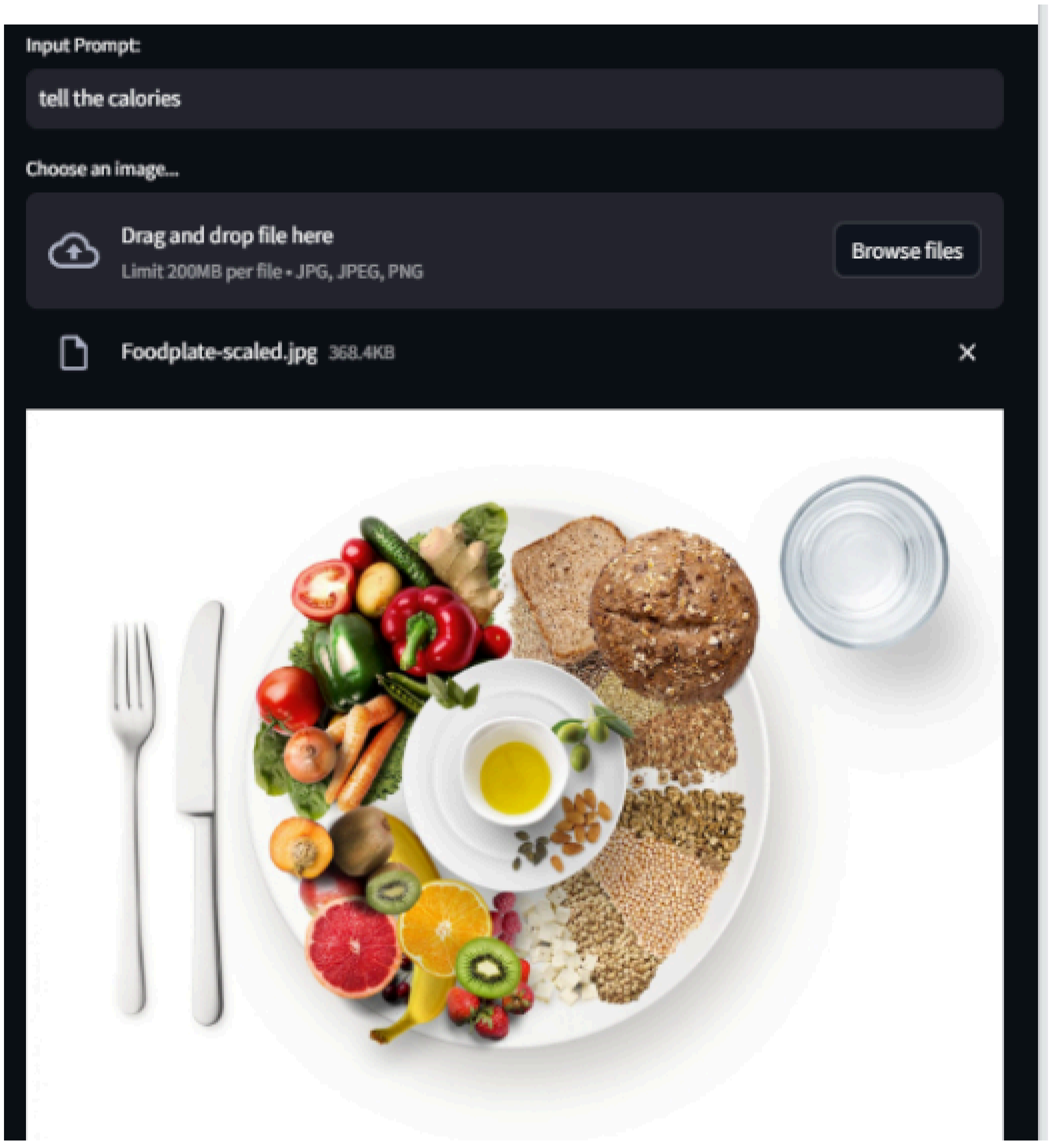


```
PS C:\Users\dhurvi patel\Desktop\SCIT academic\Smart_Bridge_Internship\SB_Company work\May_Month\Resume ATS Tracking LLM Gemini> streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.29.80:8501
```

Run the command to get the below results


Deploy
AI Nutritionist App
Input Prompt:
tell the calories
Choose an image...
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
Tell me the total calories

Input 1:



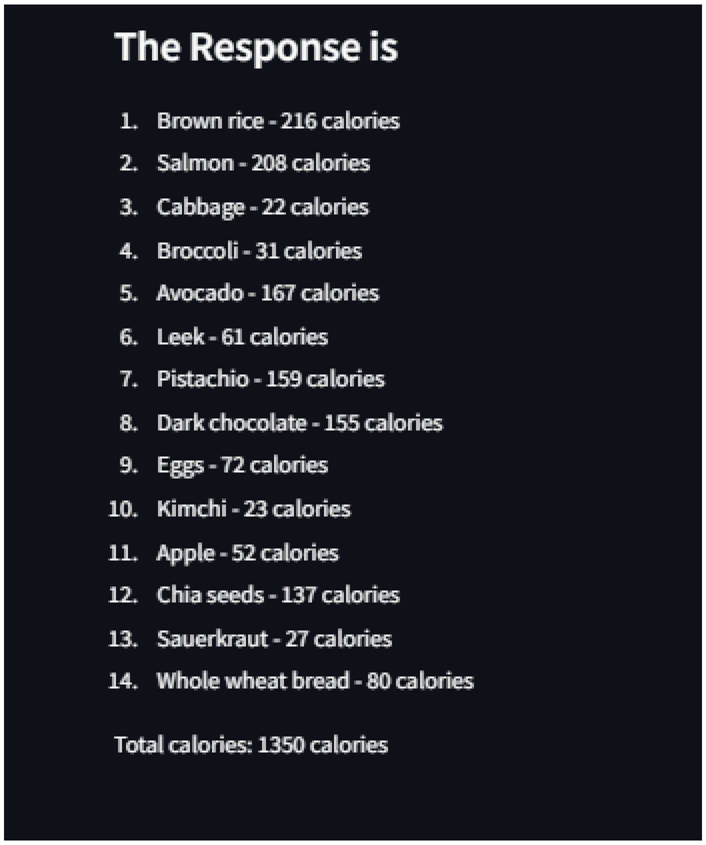
Input Prompt:

tell the calories

Choose an image...

Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
Foodplate-scaled.jpg 368.4KB

Output 1:



The Response is

1. Brown rice - 216 calories
2. Salmon - 208 calories
3. Cabbage - 22 calories
4. Broccoli - 31 calories
5. Avocado - 167 calories
6. Leek - 61 calories
7. Pistachio - 159 calories
8. Dark chocolate - 155 calories
9. Eggs - 72 calories
10. Kimchi - 23 calories
11. Apple - 52 calories
12. Chia seeds - 137 calories
13. Sauerkraut - 27 calories
14. Whole wheat bread - 80 calories

Total calories: 1350 calories